VOICE RECOGNITION - SOFTWARE SOLUTIONS IN REALTIME ATC WORKSTATIONS [1]

Alicia Lechner, St. Cloud State University, St. Cloud, MN

Kevin Ecker, Lockheed Martin, Eagan, MN [2]

Patrick Mattson, Chair Aviation Department, St. Cloud State University

Abstract

Speech recognition is a key component in the building of software for an air traffic controller (ATC) workstation used to support a Controller-Pilot Data Link Communications (CPDLC) system. In the early 1990's, developers relied on hardware to make speech recognition a reality. Limitations in processing power restricted users to small grammar files, limited vocabularies and lower accuracy recognition rates. Features desired by air-traffic controllers, such as the ability to use dynamic call signs and compound messages, became more feasible with the advances in technology.

This paper examines the challenges and opportunities of developing voice recognition software solutions in ATC workstations using multiple dialects and accents, complex and varied grammars and terminology, accuracy, hardware restrictions, user-friendly vs. functionality and user-training procedures. Voice recognition technology today is not perfect for reasons discussed throughout this paper. Therefore, consideration must be made to determine what is "tolerable". For example, for safety critical systems, which are common in the aviation field, a high degree of, if not perfect, accuracy would almost certainly be desired. In any event a level of required accuracy must be determined and efforts must be made towards achieving that accuracy.

Our work, under the direction of the Avionics Engineering Center at Ohio University, was in support of the Federal Aviation Administration's (FAA) Runway Incursion Reduction Program (RIRP) and the National Aeronautics and Space Administration (NASA) Runway Incursion Prevention System (RIPS) conducted at the Dallas-Fort Worth International Airport (DFW).

Introduction

The Controller Communication and Situation Awareness Terminal (C-CAST) is a combination input device and graphical display which is able to transmit, display, and receive clearances with aircraft through a data link channel using voice recognition and a touch-screen monitor. C-CAST provides for increased situational awareness for ground controllers including aircraft identification, position, direction and intent. The protocol for communicating between ATC and aircraft has been defined by Radio Technical Commission for Aeronautics (RTCA) in their Minimum Operational Performance Standards for ATC Two-Way Data Link Communications (DO-219) and is incorporated in the C-CAST. Figure 1 depicts the C-CAST to DLM interface which uses a TCP/IP communications protocol; the DLM is configured as a Server (port 9001) and C-CAST as Client [1].

Figure 1. Voice Recognition Configuration [1]

The system process converts voice entered controller instructions to digitized messages that are formatted according to the RTCA DO-219 standard. Pilot acknowledgments of controller messages are downlinked to the system and transferred to the C-CAST. In addition to voice recognition, messages can be entered using the monitor's touch screen or by mouse and/or keyboard.

1 This research project was funded by Ohio University via NASA-Langley Research Center, Hampton, VA

2 Kevin Ecker Graduated from St. Cloud State University, MN

A computer monitor, with touch screen capability, is used to convey information to the controller. Standard FAA-formatted aircraft data from the ARTCC database is displayed on flight strips. The flight strips are electronic versions of the strips currently used in the ATC. Outgoing messages are displayed as message text on the flight strips for reference. A map of the airport with real time traffic is displayed on the monitor. C-CAST is also capable of displaying runway hold bars and incursion alerts sent by the aircraft.

Advantages of voice recognition in ATC

Several advantages are readily apparent when considering implementing speech-recognition features in ATC applications. One consideration is that speech recognition employs a "hands-off" approach. Users speak control instructions into a headset equipped with a microphone. Because controllers are familiar with this setup, a minimum amount of time would be needed to familiarize users with the speech-recognition system. The combination of a speech-recognition engine and a graphical user interface (GUI) interface would be more efficient; the interface would provide ATCs with the plane's position on the runway, allowing for immediate corrections, if needed. Finally, speech-recognition has advanced to the point that variations in pronunciation and inflection are easily accommodated, allowing the software to be used on a broad scale in many regions.

Continuous speech recognition

Speech-recognition technology became widely available in the mid-1990's. While the concept proved promising, the hardware demands and small vocabularies severely limited its usage. Users spent precious time training with the system, creating a vocal footprint that ensured high accuracy rates with that individual only; because each user had to create their own account, early speech-recognition systems using larger vocabularies were unfeasible, time and space-wise, with large numbers of users.

As computers incorporated faster processors and larger memory caches, speech recognition became more widely used. Systems shifted from discrete speech, which recognized single words:

"NASA. 557. Request. Denied."

to continuous speech that allowed multi-word phrases and sentences:

"NASA 557 Request Denied."

While easier for individuals to use, continuous speech recognition poses special problems for us as we designed this system. Careless design would reduce accuracy rates with complex sentences and use processor and memory resources at the expense of other procedure. We felt if the process was implemented correctly then continuous speech recognition systems could improve user interaction.

Transition from hardware to software implementation of voice recognition

Originally C-CAST relied on a Verbex voice recognition card. A Verbex card was installed into the computer workstation and allowed users via voice recognition to input simple control instructions. Initially obtained in 1996, the cards proved to be extremely reliable; in the 1997 Atlanta tests, the Verbex system had a 97 percent accuracy rate [2]. While dependable, the system had limitations: Verbex only supported DOS drivers and lacked support for the Win32 API platform. The next step was to implement dynamic call signing and complex, multi-instruction messages; it became clear that the speech recognition system needed to be updated. The Verbex system also required users to train for up to one hour before using the program; while this ensured higher accuracy rates; the training period was inconvenient and required considerable time and effort in maintaining user's voice profiles.

In the fall of 1999, after careful consideration, the Lernout & Hauspie (L&H) speech recognition engine was chosen to replace the Verbex system. The L&H engine worked well with Windows-based applications and supported the large, complex grammar files that were needed. The grammar file could be easily modified to allow for multiple pronunciations of a single word or phrase, which eliminated users' training time and allowed users with discrete accents to readily use the application. The software could easily handle the demands of dynamic call signing.

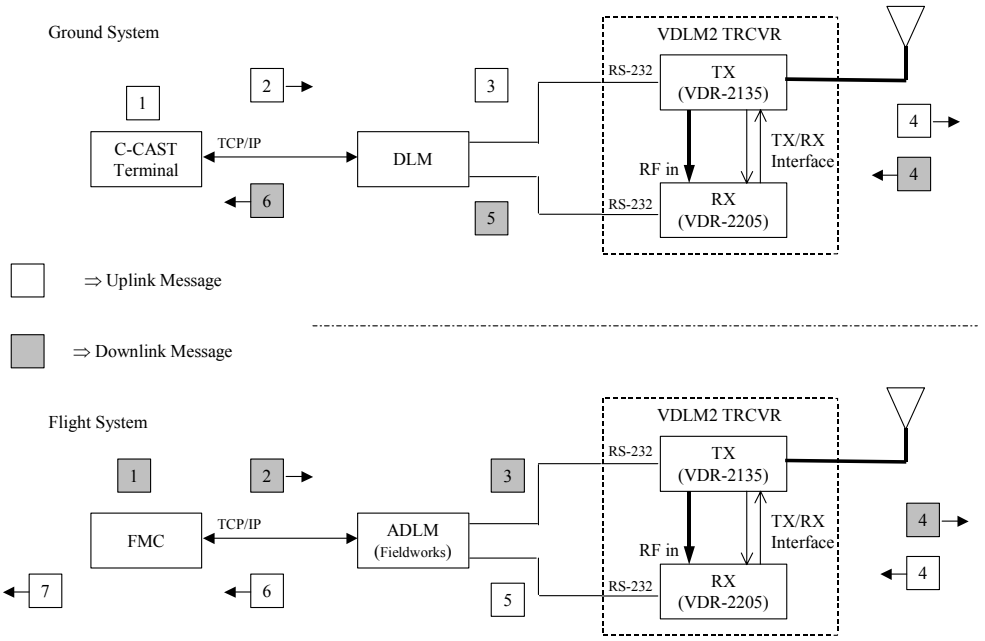

Figure 2. CPDLC Data Link System [1]

Voice recognition issues

Figure 2 shows the overall CPDLC system architecture and interfacing details [1] that we used for our blueprint for designing the software. As a relatively new technology, voice recognition still poses several challenges. While time and development may minimize or even eliminate these issues, currently they remain potential obstacles to successful implementation of voice recognition. By giving careful consideration to these issues, these "obstacles" can be overcome.

Voice recognition systems often utilize grammar files; a grammar file defines the syntax of what will be said in the use of this system. Also included are any special pronunciations, which will be discussed later. From this grammar file the voice recognition system is able to compile lexical trees (Figure 3 & 4) which the system uses to recognize a statement, by parsing the tree and matching words to the syntax defined by the tree and grammar file.

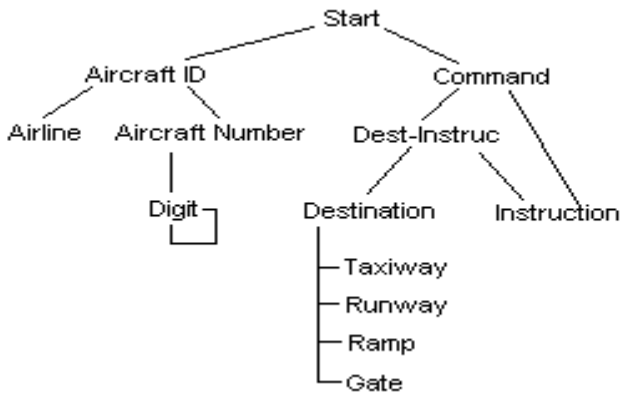

Figure 3. Partial Lexical Tree

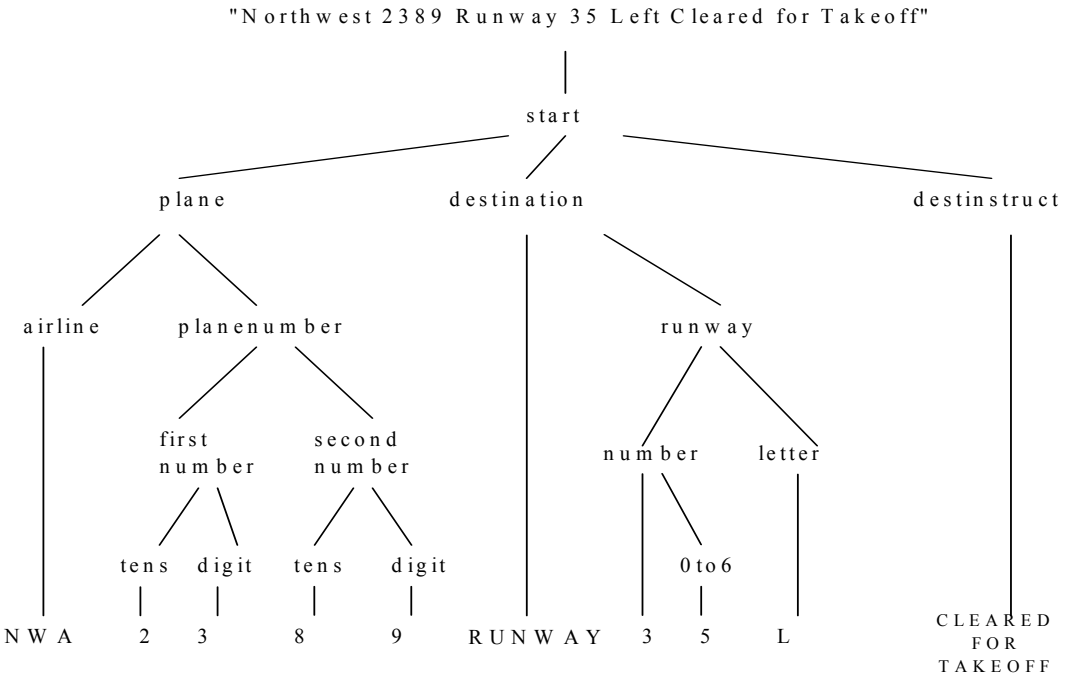

Figure 4. Completed Lexical Tree

Multiple dialects and accents

Voice recognition engines are capable of accepting and correctly interpreting most subtle differences in accent. However, for extreme variations in pronunciation, alternatives must be defined within the grammar file. For example, controllers use the term “Kaybec” rather than “Quebec” as the phonetic name for the letter “Q”; the system should return “Quebec” as the result of both pronunciations. “Kaybec" should be designated as an alternative pronunciation for "Quebec". Multiple dialects and accents are a factor in deploying an application incorporating speech recognition.

Multiple accents can also reduce accuracy; extreme variations in word pronunciation can cause a sensitive speech recognition engine to reject correct input. While individual words can be altered to accept multiple versions, redefining many words in large vocabularies might be too-time consuming. In this case, the engine accuracy rate can be reduced to accept a larger of variety of input, in situations such as ATC communications, care must be taken to ensure the accuracy of the instructions. Speech recognition engines support multiple languages only requiring the swapping of grammar files as needed.

Complex and varied grammars/terminology

One of the more difficult aspects of implementing voice recognition is the creation, refinement and maintenance of the grammar file; the file must contain all possible phrases and commands that might be uttered to the system. The terminology and layout of all possible phrases must be rigidly defined and strictly adhered to by users that includes everything from the simplest acknowledgement to the most complex taxi route command. Tables 1 and 2 show sample messages that were incorporated into the system design. [3]

Table 1. CPDLC Uplink Messages

Message #	Description
240	HOLD SHORT OF [position]
241	TAXI RUNWAY [runway] VIA [taxiroute]
242	TAXI RAMP [ramp] VIA [taxiroute]
243	CROSS [position] [WITHOUT DELAY]
244	CONTINUE TAXI

Table 2. CPDLC Downlink Messages

Message #	Description
1	UNABLE
3	ROGER
102	LANDING REPORT
202	TAXI DEVIATION
203	TURNED-OFF ON TAXIWAY [taxiway]
204	TAXI DEVIATION RESOLVED
205	RUNWAY INCURSION [source] [alarm type] [identification]
206	ASSIGNED GATE [gatenumber]

Several guidelines were used to design the system grammar. First, the grammar file should be as simple as possible. Longer phrases can be broken up into smaller, more manageable chunks. The use of "wake-up words" can help avoid errors and mangled outputs, especially in complex grammars. If one word or phrase type always precedes another, the speech recognition engine can use the former as a marker, improving recognition times. For example, when ordering an aircraft to particular piece of concrete we know that the ATC will tell the aircraft “CROSS” followed either by a Taxiway or Runway. Therefore, we can write the following in the grammar file:

<COMMAND>:
: CROSS RUNWAY <runwaynumber>
: CROSS TAXIWAY <taxiwayname>;

as opposed to:

<COMMAND>
: CROSS <location>;

Figure 5. Grammar Example

In Figure 5 <runwaynumber>, <taxiway name>, and <location> represent a multitude of further options available to the user. Runway numbers and taxiway names are formatted differently, allowing the speech recognition engine to differentiate between the two. Instead of trying to choose between every runway and taxiway available, the number of possible choices has been narrowed to two, reducing the chances of an erroneous message. In essence, we break one large statement into a multitude of smaller statements.

Grammar files should only incorporate meaningful results; if a specific range of values is used, the grammar file can exclude choices outside of the accepted range. For example, frequencies for control towers should not consider negative values. Making a grammar file accept generic values might be tempting, especially with large files, but will reduce accuracy rates and slow recognition times. Therefore, we should ensure, whenever possible, that only meaningful results are allowed for our grammar file.

Occasionally, situations will arise when a generic portion of the grammar file is needed. One example is airplane’s call sign. A list of all possible call signs would not only be extremely large (potentially wasting space) but would also present the voice recognition engine with a extreme range of options (many of which sound very similar), increasing the likelihood of inaccurate recognition.

The list of possible aircraft numbers (IDs) is broken down into several manageable pieces by using dynamic call signing,. The first part of the call sign is the airline itself. Each abbreviated airline name is matched with the user's input; if "Northwest" was spoken, the speech recognition engine would return “NWA”. The ID number is handled using known constraints. Since an airline call sign number has a maximum of known digits and each digit ranges from 0 to 9, the engine is instructed to listen for up to four digits, then truncate the four separate digits into one and add to the aircraft name to create the complete call sign. Figure 6 shows an example of user input and the corresponding output.

‘NASA five five seven”

is interpreted as:

"NASA557"

Figure 6. Dynamic call signs.

This method of simplifying potentially crippling complexity can be applied in many different situations. While it is a simple solution to a complex problem, it is not a universal one. In some cases, simply listing the possibilities will be the better option. Careful consideration should be

given for when a generic solution (such as call signing) is appropriate.

Accuracy

Maintaining accuracy rates is a significant challenge in developing speech recognition applications. Two key issues exist when considering accuracy rates; first, the precision rates must be within acceptable levels for the particular system. Additionally error-handling measures are necessary as these systems are not perfect and some errors will occur.

Error levels vary with each system's purpose. Voice dictation programs for word processors allow users to correct their mistakes; a lower degree of accuracy is needed with this software. Safety critical systems, such as ATC communications and others involving sensitive data require the highest degree of accuracy possible which should be decided upon during the system's design phase..

In addition, it is also necessary to decide what type of errors is preferred. In the recognition process, the voice recognition system can attempt to eliminate possible results that it deems incorrect. This presents a unique problem for the system designers. Is it better for the system to return slightly incorrect results or nothing at all? This decision depends heavily on the role speech recognition plays within the program.

Hardware restrictions

Speech recognition systems place significant demand on processor and memory resources; system usage is directly proportional to the size and complexity of the grammar file. Resource allocation is a significant concern when designing applications with speech recognition. C-CAST showed a marked performance decrease when tested on systems with Pentium II 300 MhZ or less as the speech recognition features interfered with the sending and receiving of messages across the TCP/IP connection. Computers using speech recognition programs must be equipped with a sound card compatible with the system and an intake device, such as a microphone.

User-friendly vs. Functionality

Software incorporating voice recognition systems are required to be user-friendly, yet capable of handling many tasks efficiently. Users should barely notice that they are using a computerized system, as a well-integrated speech recognition application will closely mimic real-life interactions. Unfortunately, even the most complete grammar file will not completely capture real-life ATC communications. By limiting the variety and formatting of the input, designers risk turning applications into basic simulators; an acceptable compromise must be reached.

Dynamic call signing properly implemented is a good example of a balance between functionality and user-friendliness. By allowing the system to determine at that moment the aircraft's call sign we are accomplishing both goals. However, care must be taken so that the system is robust enough that it maintains a zero or very low rate of errors.

The FAA's standard format for controller instructions to aircraft determines that the most common phrases begin with the aircraft ID followed by a series of instructions. Even though a single message may contain many instructions, using compound messaging allows for a large variety of messages with a high accuracy rate and minimum demands on processor time.

Compound messaging uses many of the techniques employed in dynamic call signing. In a message, the aircraft ID is followed by one or more instructions. If each command to the aircraft is represented by <instruction> in the grammar file, the top level of the grammar file would contain:

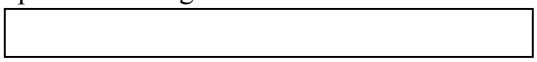

Figure 7. Aircraft instruction

In Figure 7, the aircraft is identified first, then the remainder of the message is broken down into a series of instructions. Instead of trying to identify a single large message, the engine will repeatedly decipher each instruction. At project runtime this structure is only slightly more complex than processing a single instruction, with minimal overhead in processor time involved. As has been shown, a system can be made to be user-friendly while still preserving its functionality. With the proper planning and creativity, solutions to seemingly critical problems can be found.

User-Training

Early speech-recognition systems required the user to spend significant time working with the

program to achieve higher accuracy rates. By creating their own profile, users enjoyed a high level of accuracy; however, because the sample was taken over a single training session, the quality and timbre of the user's voice was so narrow that very little variation was allowed. If the individual had a cold or other condition that changed their voice, the user profile would be useless, requiring further training or tinkering with the system to reduce accuracy.

Recent advances in technology have allowed speech-recognition engines to become more powerful and sensitive, returning correct values in a multitude of environments. User-training is now optional and only necessary under two conditions: if a word or phrase is consistently misinterpreted by the engine, or if the user's voice or accent is so remarkable that a unique file needs to be created. If users are having difficulty, they simply might require more time; as users became familiar with C-CAST, their accuracy rates and comfort with using the system improved significantly.

Conclusion

The Controller Communication and Situation Awareness Terminal (C-CAST), tested at the Dallas-Ft. Worth International (DFW) airport, is a good example of how speech recognition can greatly enhance the degree of interaction between users and software. Technological advances have allowed individuals greater flexibility and faster performance when working with applications. While some additional hardware is still necessary, speech recognition packages are much more affordable cost- and resource-wise. From the successful tests in Atlanta (1997) to the addition of dynamic call signs and compound messages at DFW (2000), speech recognition has been an integral part of this project.

The voice recognition system performed well even though the ambient noise in the room sometimes was loud. The grammar file was generic with minimal tailoring to the specifics of the Dallas-Ft. Worth airport to enable the team to test dynamic phrasing. The C-CAST was successful at providing the NASA 757 with taxi route information via Controller-Pilot Datalink Communications. The requirement to use standard phraseology proved to be the main shortcoming with the voice recognition system as implemented. Air traffic controllers as well as other visitors during the tests and demonstrations thought that this type of system could enhance surface operations at a variety of airports.

Acknowledgements

The authors wish to acknowledge the help and support given to them during this project. Lead researcher Dr. James Rankin, Director Avionics Engineering Center, Ohio University, Athens, OH, coordinated our efforts and shared his experience with encoding protocols. Also, fellow project members Eric Best, Sanjeev Gunwardeen and Qingwei Ma were a constant source of support and inspiration.

References & Works consulted

[1] Rankin, J. July 18, 2000. "Controller – Communication and Situational Awareness Terminal (C-CAST) to Data Link Manager (DLM) Interface Control Document" Revision 1.1. Avionics Engineering Center Ohio University, Athens, OH.

[2] Rankin, J., and Mattson, P., October 1997. "Controller Interface for Controller-Pilot Data Link Communications", Proceedings of the 16th DASC.

[3] Rankin, J. July 31, 2000. "C-CAST Message Set for DFW RIPS Demonstration, Revision 7".

"Minimum Operational Performance Standards for ATC Two-Way Data Link Communications", RTCA DO-219.